week 4 design exercise

Tasks

1. Write 250 words that critically reflect on this week's reading:
   Malpass, M. 2017. Practice. Critical Design in Context. London: Bloomsbury, pp.91-122.
2. Create a theremin using an ultrasonic sensor and a piezo electric buzzer.

Key Terms

`library`

A library is a file with additional code that can provide extra functionality to an Arduino. Often sensors, motors, and extra hardware require libraries to function properly. Libraries are often downloaded as .zip files and can be installed in the Arduino IDE by selecting from the top toolbar: `Sketch > Include Library > Add .ZIP Library...`

`function`

A function is a piece of code that performs a specific task. Arduino contains standard functions like `digitalWrite(somePin, HIGH);` which will turn a pin on. However, you can create custom function for new tasks. This allows you to create modular code with defined tasks.

`map(value, ALow, AHigh, ZLow, ZHigh)`

Map is a function that 're-maps', or proportionally changes, one set of values to another. In otherwords a `value` of `ALow` would output as `ZLow`. A `value` of `AHigh` would output as `ZHigh`. All the values in-between would be proportionately output from `ZLow` to `ZHigh`.

Ex: To proportionally mapp a variable with a value 5 to output proportionally between 100 and 1000, you would require the following code:

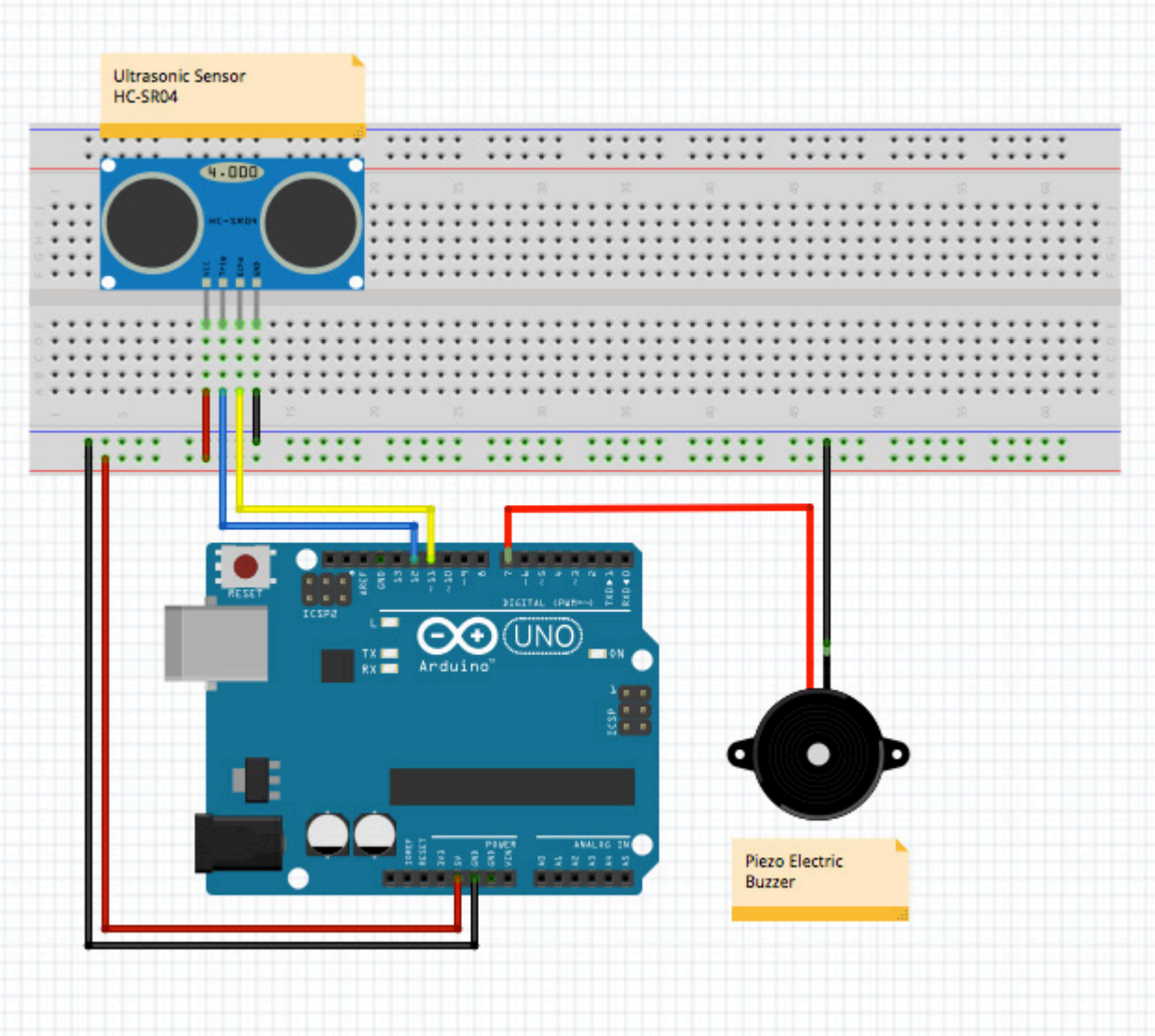
```
int someVariable = 5;          // the variable 'someVariable' has a value of 5
int mappedVariable;            // create new variable called 'mappedVariable'

// store a new mapped value in variable 'mappedVariable' that proportionally
   'maps' the value of variable named 'someVariable' in a set from 1 and to 10
   to a set between 100 and 1000.
mappedVariable = map(someVariable, 1, 10, 100, 1000);
```

The output of `mappedVariable` is `500`.

Wiring Setup – Ultrasonic Sensor:

The ultrasonic sensor detemines the distance of an object by emitting ultrasonic sound from the trigger and collecting the bouncing wave in the receiver. Using this sensor, create a theremin that changes tone in response to how close your hand is to the sensor.

First, connect the ultrasonic sensor (part no: HC-SR04) to the Arduino using on the diagram below. You will need to bend your sensor so it faces upward. Then connect the piezo electric buzzer to the arduino as show below (note: the colours of the wires in diagram may not be the same as the colours of your wires.)



Programming - Ultrasonic Sensor

Now, create two constant values for the ultrasonic sensor. The first will be the `TRIGGER_PIN` (pin 12), which will send out an ultrasonic sound. The second will be the receiver, the `ECHO_PIN` (pin 11), which collects the reflected ultrasonic sound to determine the distance of the object.

```
#define TRIGGER_PIN   12
#define ECHO_PIN      11
```

Next, define two variables. The first will be a long integer called `duration` which will store the length of time it takes for the sensor to send and receive the reflected ultrasonic sound. The second will be an integer called `distance` which will store the distance value:

```
long duration;    //variable to store time to receive ultrasound
int distance;     //variable to store distance of object
```

Then, in the `setup()` function, initialise your arduino pin variables. `TRIGGER_PIN` (pin12) will be set to `OUTPUT`. `ECHO_PIN` (pin11) will be set to `INPUT`. Then initialise your Serial Monitor so you can print out the distance values on your computer.

```
void setup() {
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.begin(9600);
}
```

Then, in the `loop()` function, include the following code which will: (1) clear, or turn off, the `TRIGGER_PIN` for 2 microseconds; (2) send out ultrasound via the `TRIGGER_PIN` for 10 microseconds and (3) turn the `TRIGGER_PIN` off.

```
void loop() {
  // Clears the TRIGGER_PIN
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);

  //Sets the TRIGGER_PIN on HIGH state for 10 micro seconds
  digitalWrite( TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite( TRIGGER_PIN, LOW);
```

Then, include the following code which will collect the reflected ultrasound via the `ECHO_PIN` and store the value in the variable `duration`.

```
  // Reads the ECHO_PIN, returns the sound wave travel time in microseconds
  duration = pulseIn(ECHO_PIN, HIGH);
```

Then write the following code to convert the duration value into a distance value in centimetres:

```
  // Calculates cm based on duration of ultrasound from trigger to receiver
  distance = duration*0.034/2;
```

Then, include the following code to print the distance value to the Serial Monitor:

```
  // prints the distance in the Serial Monitor
  Serial.print("distance: ");
  Serial.print(distance);
  Serial.println("cm");

}
```

Programming - Ultrasonic Sensor

The code for the ultrasonic sensor should look like the code below. Upload this sketch to your Arduino, then select from the top toolbar: `Tools > Serial Monitor`. The Serial Monitor dialog box will appear. It should print out the values from the sensor every 50ms.

```
#define TRIGGER_PIN   12
#define ECHO_PIN      11

long duration;    //variable to store time to receive ultrasound
int distance;     //variable to store distance of object

void setup() {
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.begin(9600);
}

void loop() {
  // Clears the TRIGGER_PIN
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);

  //Sets the TRIGGER_PIN on HIGH state for 10 micro seconds
  digitalWrite( TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite( TRIGGER_PIN, LOW);

  // Reads the ECHO_PIN, returns the sound wave travel time in microseconds
  duration = pulseIn(ECHO_PIN, HIGH);

  // Calculates cm based on duration of ultrasound from trigger to receiver
  distance = duration*0.034/2;

  // prints the distance in the Serial Monitor
  Serial.print("distance: ");
  Serial.print(distance);
  Serial.println("cm");

}
```

Programming - Piezo electric buzzer

Create a new sketch and write the following code, which will create a tone for your buzzer.

```
int tonePin = 7;

void setup() {

}

void loop() {
  tone(tonePin, 200);   // tonePin will generate a freq 200hz
}
```

Upload your sketch to your arduino. Double-check that your buzzer is creating a sound.

Programming - Sensor + Buzzer

Once the sensor and the buzzer are functioning individually, it is time to combine them. First, copy your variable from your buzzer sketch and paste it into your sensor sketch. Then, copy the code from the `setup()` function in your buzzer sketch and paste it into the `setup()` function of your sensor sketch. Then, copy the code from the `loop()` function in your buzzer sketch and paste it into the `loop()` function of your sensor sketch. It should look like the following:

```
#define TRIGGER_PIN   12
#define ECHO_PIN      11

int tonePin = 7;  // variable tonePine assigned to pin 7
long duration;    //variable to store time to receive ultrasound
int distance;     //variable to store distance of object

void setup() {
  pinMode(tonePin, OUTPUT);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.begin(9600);
}

void loop() {
  // Clears the TRIGGER_PIN
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);

  //Sets the TRIGGER_PIN on HIGH state for 10 micro seconds
  digitalWrite( TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite( TRIGGER_PIN, LOW);

  // Reads the ECHO_PIN, returns the sound wave travel time in microseconds
  duration = pulseIn(ECHO_PIN, HIGH);

  // Calculates cm based on duration of ultrasound from trigger to receiver
  distance = duration*0.034/2;

  tone(tonePine, 200);

  // prints the distance in the Serial Monitor
  Serial.print("distance: ");
  Serial.print(distance);
  Serial.println("cm");

}
```

Programming - Sensor + Buzzer

Then, create a new variable called `mappedTone`. Then, inside the `loop()` function, above the `tone()` function, write a `map()` function that remaps the value of your `distance` variable to a tone frequency between 0 and 2000hz. Store this remapped value in your new variable called `mappedTone`. Then, change the `tone()` function so the output tone is not 200hz, but instead is the variable called `mappedTone`. Your code should look like the code below:

```
#define TRIGGER_PIN   12
#define ECHO_PIN      11

int tonePin = 7;  // variable tonePine assigned to pin 7
long duration;    //variable to store time to receive ultrasound
int distance;     //variable to store distance of object
int mappedTone;

void setup() {
  pinMode(tonePin, OUTPUT);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.begin(9600);
}

void loop() {
  // Clears the TRIGGER_PIN
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);

  //Sets the TRIGGER_PIN on HIGH state for 10 micro seconds
  digitalWrite( TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite( TRIGGER_PIN, LOW);

  // Reads the ECHO_PIN, returns the sound wave travel time in microseconds
  duration = pulseIn(ECHO_PIN, HIGH);

  // Calculates cm based on duration of ultrasound from trigger to receiver
  distance = duration*0.034/2;
  mappedTone = map(distance,0,200,0,2000);

  tone(tonePin, mappedTone);

  // prints the distance in the Serial Monitor
  Serial.print("distance: ");
  Serial.print(distance);
  Serial.print("cm, Frequency: ");
  Serial.print(mappedTone);
  Serial.println("hz");

}
```

Upload your code and then test your theremin.

Record your video

Record a 30-second video that shows:

1. your hardware connection
2. your final sketch
3. the Serial Monitor output of your final sketch
4. the functioning system

Create a .zip file that contains your 30-second video and your written text about this week's reading. Upload your files to the dropbox link which will be provided at the start of next class. You cannot receive credit for this assignment if you are not present in class.